
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**
**BRNO UNIVERSITY OF TECHNOLOGY**

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**
**FACULTY OF INFORMATION TECHNOLOGY**

**ÚSTAV INTELIGENTNÍCH SYSTÉMŮ**
**DEPARTMENT OF INTELLIGENT SYSTEMS**


# NÁSTROJ PRO TVORBU CEG GRAFŮ

**A TOOL FOR BUILDING CEG GRAPHS**

**BAKALÁŘSKÁ PRÁCE**
**BACHELOR'S THESIS**


**AUTOR PRÁCE** — **FILIP MUTŇANSKÝ**
**AUTHOR**

**VEDOUCÍ PRÁCE** — **Ing. ALEŠ SMRČKA, Ph.D.**
**SUPERVISOR**



**BRNO 2017**

# Abstrakt

Testovanie pomocou grafov príčin a dôsledkov je jeden s prístupov ku testovaniu na základe požiadaviek. Grafy príčin a dôsledkov (angl. Cause-Effect Graph, CEG) sa používajú na vizualizáciu logických vzťahov medzi požiadavkami a na vyplnenie rozhodovacej tabuľky na vytvorenie testovacích prípadov. Cieľom tejto bakalárskej práce je navrhnúť a implementovať nástroj na vytváranie a manipuláciu s grafmi príčin a dôsledkov. Vytvorený nástroj je implementovaný ako single-page webová apikácia. Manipulácia s grafom je možná cez grafické rozhranie, alebo cez textovú definíciu grafu v jazyku vytvorenom pre CEG. Nástroj používa logic solver na uľahčenie vytvárania rozhodovacej tabuľky užívateľovi.



# Abstract

Cause-effect graphing is one of approaches to requirement-based testing. Cause-effect graphs (CEG) are used to visualize logic relations among requirements and to fill in a decision table to form test cases. The goal of this bachelor's thesis is to design and implement a tool for creating and manipulating with CEG graphs. The developed tool is implemented as a single-page web application. The manipulation with CEG is possible via graphical user interface and/or via text-based specification in a CEG specific language. Moreover, the tool uses logic solver to ease a user with creation of the decision table.



# Klíčová slova

testovanie na základe požiadaviek, Javascript, D3.js, CEG, rozhodovacia tabuľka

# Keywords

requirement based testing, Javascript, CEG, decision table


# Citace

MUTŇANSKÝ, Filip. *Nástroj pro tvorbu CEG grafů*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Aleš Smrčka, Ph.D.

# Nástroj pro tvorbu CEG grafů

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pána Ing. Aleša Smrčky, PhD. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Filip Mutňanský
18. mája 2017

## Poděkování

Rád by som sa poďakoval svôjmu vedúcemu práce Alešovi Smrčkovi, PhD. za cenné rady pri tvorbe práce.

# Obsah

# 1 Úvod

Viac ako polovica chýb v softwarových projektoch bolo zavedených vo fáze špecifikácie požiadaviek [3]. Náklady na odstránenie chýb spôsobených nesprávnou, alebo nejednoznačnou špecifikáciou sú väčšie ako pri chybách v zdrojovom kóde. Preto bolo zavedené testovanie na základe požiadaviek, ktorého cieľom je znížiť nejednoznačnosť špecifikácie.

Grafy príčin a dôsledkov (angl. Cause-Effect Graph, CEG) sú jednou z metód testovania na základe požiadaviek. Pomocou tejto techniky je možné overiť správnosť, jednoznačnosť špecifikácie a vytvoriť sadu testovacích prípadov, ktorá overí, či testovaný software splňuje tieto požiadavky. Táto práca sa zaoberá návrhom a implementáciou nástroja na vytváranie a manipuláciu s grafmi príčin a dôsledkov (ďalej ako *ceg-editor*). CEG znázorňuje logické vzťahy medzi požiadavkami testovaného systému. CEG sa skladá z príčin (*causes*), dôsledkov (*effects*), obmedzení, medziuzlov a vzťahov medzi nimi. Po transformácií CEG do rozhodovacej tabuľky jednotlivé stĺpce predstavujú testovacie prípady.

## 1.1 Motivácia

Hlavnou motiváciou pre vznik nástroja *ceg-editor* je fakt, že v súčasnosti neexistuje nástroj pre tvorbu CEG grafov, ktorý by nebol proprietárny. Jediný dohľadateľný nástroj, ktorý využíva CEG na tvorbu testovacích prípadov je proprietárny software *BenderRBT* [1]. Nástroj *ceg-editor* je open-source single-page webová aplikácia (aplikácia, ktorá vyžaduje len jednu HTML stránku).

## 1.2 Popis obsahu kapitol

Popis testovania softwaru sa nachádza v kapitole 2. V kapitole 3 sú vymenované požiadavky a popísaný návrh nástroja *ceg-editor*. V kapitole 4 sú popísané technológie súvisiace s nástrojom *ceg-editor* konkrétne HTML, CSS, Javascript, Markdown, SVG a D3.js, súčasťou tejto kapitoly sú aj detaily o implementácií automatického generovania testovacích prípadov. V kapitole 4 sú na konci kapitoly vymenované nedostatky tohto nástroja s navrhnutými riešeniami. V poslednej kapitole 5 je zhrnutý prínos práce a spomenuté možné rozšírenia. V Prílohe A je popísané testovanie nástroja *ceg-editor*, ktoré prebiehalo pomocou samotného vytvoreného nástroja metódou testovania na základe požiadaviek. V prílohe B je príklad vygenerovaného reportu vo formáte Markdown.

# 2 Testovanie softwaru

Testovanie [2] je vo všeobecnosti spúšťanie programu za účelom nájdenia chyby v testovanom programe. Keby sme chceli otestovať každú možnosť, ktorá v programe môže nastať, tak je počet testovacích prípadov veľmi vysoký, alebo nekonečný. Preto je potrebné nájsť podmnožinu všetkých testovacích prípadov, ktorá má najväčšiu pravdepodobnosť na nájdenie chyby. Existujú dva prístupy k testovaniu: *black-box testing* a *white-box testing*. *White-box testing* je testovanie na základe znalosti vnútornej štruktúry softwaru.

## 2.1.1 Black-box testovanie

Black-box testovanie [2] (tiež označované ako *data driven testing*, alebo *input/output driven testing*) je testovanie bez znalosti vnútornej štruktúry softwaru. Všetky testovacie prípady sú vytvorené na základe špecifikácie. Cieľom tohto testovania je nájsť okolnosti v ktorých sa program nebude správať podľa špecifikácie, alebo podľa toho, alebo ako by užívateľ očakával.

V black-box testovaní sú využívané štyri prístupy: *rozdelenie na ekvivalenčné triedy*, *analýza hraníc*, *cause-effect graphing* a *error guessing*. Posledný prístup *error guessing* je založený na odhadovaní miest kde by testovaná aplikácia mohla mať nedostatky.

### 2.1.1.1 Rozdelenie na ekvivalenčné triedy

Pri hľadaní správnej podmnožiny testovacích prípadov je vhodné rozdeliť testovaný systém do kategórií (tried) tak, že správanie systému v jednej kategórií je pre podobné vstupy rovnaké. Jednotlivé kategórie musia byť disjunktné a musia pokrývať celé správanie systému. Pri tvorbe testovacích prípadov je potom vybraný z každej kategórie jeden prípad reprezentujúci danú kategóriu. Takýmto spôsobom je pokrytá každá kategória. Ak nejaký testovací prípad z určitej triedy nájde chybu, predpokladá sa, že každý prípad patriaci do danej triedy odhalí tú istú chybu. Tento prístup výrazne znižuje počet potrebných prípadov.

Príklad: rozsah povolených hodnôt je 1 až 50. Tento systém je možné rozdeliť na tri ekvivalenčné triedy. Prvá kategória je 1 < hodnota < 50, ďalšie by predpokladali nevalidný vstup a to: hodnota < 1 a hodnota > 50.

### 2.1.1.2 Analýza hraničných hodnôt

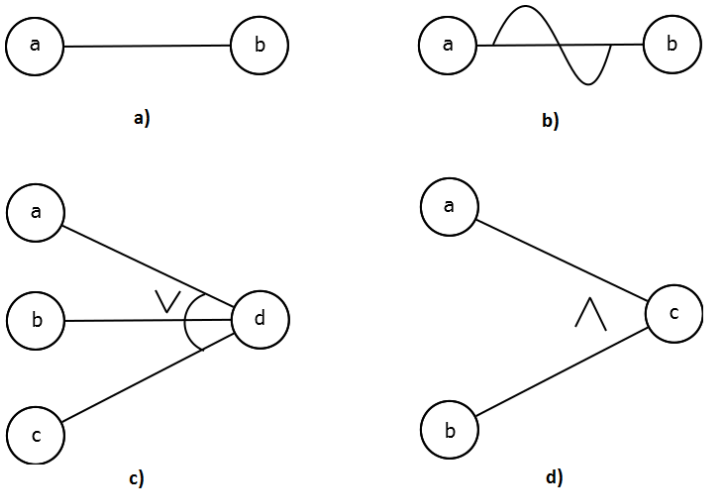
Výskyt chýb je častejší pri testovacích prípadoch okolo hraničných okolností, ako v prípadoch mimo hraníc [2]. Hraničné okolnosti sú prípady tesne nad, alebo tesne pod okrajom ekvivalenčných tried. Namiesto vybrania ľubovoľného reprezentanta ekvivalenčnej triedy analýza hraničných hodnôt vyžaduje, aby boli vybrané, také testovacie prípady, ktoré pokryjú hranice ekvivalenčných tried. Tieto hranice sa môžu týkať vstupov, ale aj výstupov systému.

Ak by bol rozsah povolených hodnôt 1 až 50. Pri využití analýzy hraničných hodnôt by boli vybrané prípady pre 0, 1, 50, 51. Ak by určitý systém vždy produkoval určitý počet záznamov, ale nikdy viac ako päť, tak by bolo vhodné vybrať také testovacie prípady, ktoré by generovali 0,1,5 a pokúsiť sa vygenerovať šesť záznamov.

#### 2.1.1.3 Cause-effect graphing

Nevýhodou predchádzajúcich metód je, že neobsiahnu kombinácie vstupov. Aj keď je testovaný systém rozdelený na ekvivalenčné triedy, tak počet kombinácií vstupov je priveľmi vysoký. Využitie CEG umožňuje systematické vybranie podmnožiny testovacích prípadov s vysokou pravdepodobnosťou nájdenia chyby. Ďalšia výhoda CEG je odhalenie nejednoznačností a chýb v špecifikácií. Nevýhodou CEG je neschopnosť znázorniť časovú následnosť udalostí, preto sa používa len pre statický popis. CEG graf je v podstate preložená špecifikácia požiadaviek z prirodzeného jazyka do boolean grafu. Proces tvorby testovacích prípadov zo špecifikácie je nasledovný [4]:

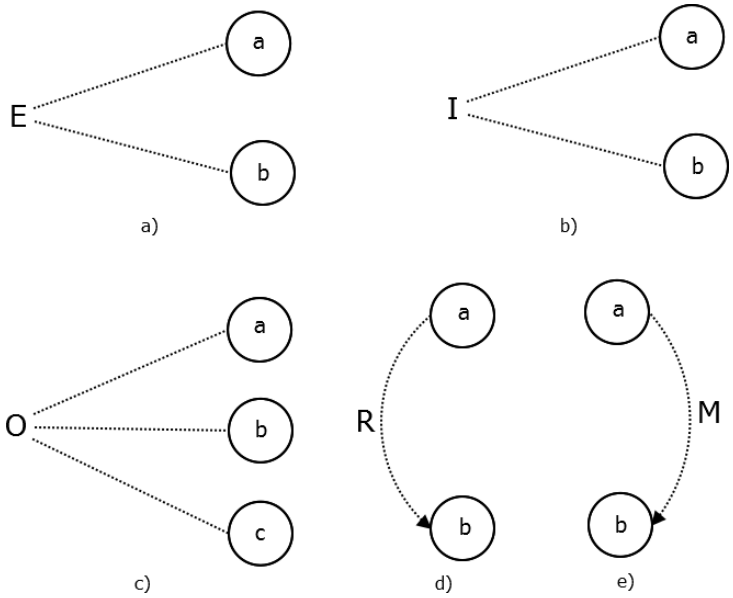
1. Rozdelenie špecifikácie požiadaviek na uzavreté časti.
2. Určenie príčin a dôsledkov.
3. Transformácia špecifikácie do CEG.
4. Určenie obmedzení nad príčinami a dôsledkami.
5. Transformácia CEG do rozhodovacej tabuľky.
6. Stĺpce tabuľky reprezentujú testovacie prípady.




Obrázok 2.1: Vzťahy v CEG. a) identita, b) negácia, c) operácia OR, d) operácia AND

Na obrázku 2.1 je znázornená syntax CEG. Uzol môže mať hodnotu 1, alebo 0. Prípad *a)* znázorňuje identitu, to znamená, že keď *a* je rovné 1, tak aj *b* bude rovné 1 a ak *a* je rovné 0, tak aj *b* bude 0. Možnosť *b)* predstavuje negáciu. V prípade, že je *a* rovné 1, tak *b* sa rovná 0, ak je *a* rovné 0, tak *b* je rovné 1. Možnosť *c)* je operácia *or*, ak je platný aspoň jeden z uzlov *a, b, c*, tak je platný aj uzol *d*. Posledná možnosť *d)* znázorňuje operáciu *and*, ak sú platné oba uzly *a* a *b*, tak bude platný aj uzol *c*, v inom prípade je uzol *c* neplatný.

Pri tvorbe CEG grafov je niekedy potreba obmedziť kombinácie príčin, alebo dôsledkov. Pre tento dôvod boli zavedené obmedzenia. Ich syntax je znázornená na obrázku 2.2. Medzi obmedzenia príčin patrí e*xclusive* (E), i*nclusive* (I), o*ne* (O) a r*equires* (R). Pri *exclusive* platí, že žiadny, alebo jeden z *a,b* je platný (platí najviac jeden uzol). Pri *inclusive* platí, že platí aspoň jeden uzol z *a,b*. Pri




Obrázok 2.2: Obmedzenia v CEG grafoch

*one* platí, že je platný práve jeden z uzlov *a,b,c*. Pri *requires* musí byť platné *a* aby mohlo byť platné *b* (nesmie nastať situácia, kedy by bolo platné *a* a nebolo by platné *b*). Medzi obmedzenia dôsledkov patrí *masks*. Pri *masks* platí, že ak je platné *a* tak *b* je nútene neplatné.

Pre demonštráciu tvorby grafu je k dispozícií názorná ukážka špecifikácie [2].

Znak v stĺpci 1 musí byť buď "A", alebo "B". Znak v stĺpci 2 musí byť číslica. Ak sú splnené predchádzajúce predpoklady, tak je súbor aktualizovaný. Ak je prvý znak nevalidný, tak je vypísaná správa X12. Ak druhý znak nie je číslica, tak je vypísaná správa X13.

Prítomné príčiny sú:

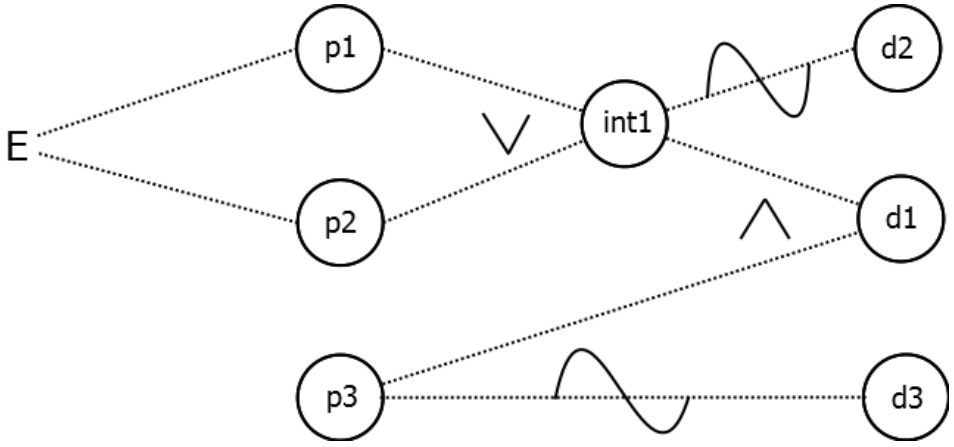
- p1 - znak v stĺpci 1 je "A"
- p2 - znak v stĺpci 1 je "B"
- p3 - znak v stĺpci 2 je číslica.

Dôsledky sú:

- d1 - súbor je aktualizovaný
- d2 - je vypísaná správa X12
- d3 - je vypísaná správa X13

V systéme nemôže nastať situácia kedy by bol znak v stĺpci 1 "A" a súčasne "B". Preto zavedieme nad príčinami *p1* a *p2* obmedzenie *exclusive*. Výsledný graf je znázornený na obrázku 3. Po operácií *or* s uzlami *p1* a *p2* vznikol medziuzol *int1*.

Následne je potrebné z výsledného grafu vytvoriť rozhodovaciu tabuľku. Nie je nutné vymenovať všetky kombinácie príčin a dôsledkov, záleží na kritérií pokrytia. V riešení v tabuľke č.1 je dosiahnuté pokrytie dôsledkov aj príčin. To znamená, že všetky dôsledky aj príčiny sa vyskytujú v oboch stavoch (T - platný, F - neplatný).




Obrázok 2.3: Graf príčin a dôsledkov vytvorený zo špecifikácie

| Príčiny | [1] | [2] | [3] | [4] | [5] |
|---|---|---|---|---|---|
| p1 | T | T | F | F | F |
| p2 | F | F | F | T | T |
| p3 | T | F | F | F | T |
| Dôsledky | | | | | |
| d1 | T | F | F | F | T |
| d2 | F | F | T | F | F |
| d3 | F | T | T | T | F |

Tabuľka 2.1: Rozhodovacia tabuľka pre graf z obrázku 2.3

### 2.1.2 Requirement based testing

Podľa [3] až 56% chýb v softwarových projektoch bolo zavedených už vo fáze tvorby požiadaviek. Okolo 50% z týchto chýb sú výsledkom nejasných, nejednoznačných a chybných požiadaviek. Zvyšných 50% chýb z fázy tvorby požiadaviek sú kvôli nekompletnosti špecifikácie [3] Čím skôr je chyba odhalená, tým je cena za nápravu menšia [6] Pokiaľ je chyba v zdrojovom kóde, tak je oprava jednoduchá. Ak je chyba spôsobená zlou špecifikáciou požiadaviek a je zachytená neskoro, tak je potrebné zmeniť návrh, zdrojový kód, testy a dokumentáciu.

Cieľom testovania na základe požiadaviek je overenie, že požiadavky sú správne, kompletné, jednoznačné a logicky konzistentné a vytvorenie testovacích prípadov, ktoré overia, že testovaný software plní požiadavky [6]. Primárnym cieľom CEG je overenie, že testovaný software splňuje požiadavky.

# 3 Návrh nástroja ceg-editor

V tejto časti je priblížený návrh nástroja *ceg-editor*. V kapitole 3.1 sú popísané požiadavky na nástroj *ceg-editor*. V kapitole 3.2 je popísaný návrh textovej definície. Reprezentácia grafu v podobe dát je popísaná v kapitole 3.3. Popis vykresleného grafu a manipulácie s ním je prítomný v kapitole 3.4. Všeobecný princíp fungovania je v kapitole 3.5. Správanie tlačidiel je popísané v kapitole 3.6. V kapitole 3.7 sa nachádza návrh rozhodovacej tabuľky. Návrh výstupov nástroja *ceg-editor* je v kapitole 3.8.

## 3.1 Požiadavky na nástroj ceg-editor

Cieľom práce je vytvoriť nástroj na vytváranie a editovanie grafov príčin a dôsledkov. Funkcionálne požiadavky na tento nástroj sú:

- Podpora textovej definície grafu vrátane definície príčin, dôsledkov a obmedzení.
- Interaktívne pridávanie uzlov, vzťahov a obmedzení cez graf.
- Interaktívne mazanie uzlov, vzťahov a obmedzení cez graf.
- Interaktívna editácia uzlov.
- Aktualizovanie textovej definície podľa zmien vykonaných cez graf
- Interaktívna tvorba rozhodovacej tabuľky.
- Informovanie užívateľa o pokrytí príčin a dôsledkov.
- Import a Export grafov.
- Report s vygenerovanými testovacími prípadmi a textovou definíciou grafu.

Nefunkcionálne požiadavky sú:

- Adaptácia nástroja na rôzne rozlíšenia.
- Funkčnosť na najpoužívanejších prehliadačoch (Google Chrome, Mozilla Firefox).
- Funkčnosť na dotykových displejoch.

## 3.2 Textová definícia

Textová definícia sa skladá zo štyroch častí:

- definícia príčin,
- definícia dôsledkov,
- definícia pravidiel,
- definícia obmedzení.

Na obrázku 3.1 je znázornený popis jazyka textovej definície CEG rozvinutou Backusovou–Naurovou formou. Pravidlo `ceg` značí, že jednotlivé časti musia byť uvedené v určitom poradí. Najprv musí byť uvedená definícia príčin `causes`, potom definícia dôsledkov `effects` a zoznam pravidiel `rules`. Definícia obmedzení `constraints` je nepovinná. Pravidlo `sep` znamená oddeľovač medzi časťami v podobe minimálne jedného znaku konca riadku. Voľné riadky sú ignorované. Definície príčin, alebo dôsledkov začínajú reťazcom "`Causes:`" v prípade príčin, alebo reťazcom "`Effects:`" v prípade dôsledkov nasledovaným minimálne jedným znakom konca riadku. Jednotlivé definície uzlov `node_def` sú v tvare `id: text`, pričom v `id` sú môže

obsahovať všetky alfanumerické znaky a znak '_' a v `text` sú povolené všetky nebiele znaky okrem znaku '/'. Znak '/' je rezervovaný pre komentáre. Definícia pravidiel začína reťazcom "`Rules:`" oddeleným minimálne jedným znakom konca riadku. Pravidlá sú definované v tvare `id = expr`, pričom `expr` sa líši typom operácie. Pre operácie AND a OR `expr` obsahuje jednotlivé `id` oddelené reťazcom "`&&`" ak ide o AND a "`||`" v prípade OR. Pre operáciu NOT `expr` obsahuje znak "`!`" nasledovaný jedným `id`. Ak ide o operáciu EQ, tak `expr` obsahuje iba jedno `id`. Definícia obmedzení začína reťazcom "`Constraints:`" nasledovaným minimálne jedným znakom konca riadku. Definície jednotlivých obmedzení sa líšia od ich typu. Pre obmedzenie typu *one* je definícia "`O:`" nasledovaná jednotlivými `id` oddelenými znakom ','. Obmedzenie *inclusive* začína "`I:`" a nasledujú `id` oddelené znakom ','. Typ obmedzenia *exclusive* má definíciu "`E:`" a potom `id` oddelené znakom ','. Definícia r*equires* obmedzenia je `id` po ňom reťazec "`->`" nasledovaný ďalším `id`. Obmedzenie *masks* je definované ako `id` nasledované reťazcom "`->`" a ďalším `id`.

```
ceg = causes sep effects sep rules [ sep constraints ];
sep = "\n" {"\n"};
causes = "Causes:" sep { node_def };
effects = "Effects:" sep { node_def };
node_def = id ":" text sep;
rules = "Rules:" sep { rule_def };
rule_def = id "=" expr sep;
expr = expr_and | expr_or | expr_not | expr_eq;
expr_and = id "&&" id {"&&" id};
expr_or = id "||" id {"||" id};
expr_not = "!" id;
expr_eq = id;
constraints = "Constraints:" sep { constraint_def};
constraint_def = ( oneof | incl | excl | req | masks ) sep;
oneof = "O: " id "," id { "," id };
incl = "I: " id "," id { "," id };
excl = "E: " id "," id { "," id };
req = id "->" id;
masks = id "masks" id;
id = valid_id_char {valid_id_char};
valid_id_char = "a".."z" | "A".."Z" | "0".."9" | "_";
text = valid_text_char {valid_text_char};
valid_text_char = all characters - '/';
all characters = ? all visible characters ? ;
```

Obrázok 3.1: Popis jazyka textovej definície CEG pomocou EBNF

Biele znaky medzi jednotlivými časťami sú ignorované. Textová definícia podporuje aj jednoriadkové komentáre. Komentár začína dvojicou znakov "`//`" a končí znakom konca riadku.

```
Causes:
//id: text
p1: Prvý znak je 'A'
p2: Prvý znak je 'B'
p3: Druhý znak je číslica

Effects:
d1: súbor je aktualizovaný
d2: správa X12
d3: správa X13

Rules:
int1 = p1 || p2
d2 = !int1
d1 = int1 && p3
d3 = !p3

Constraints:
E: p1, p2
```

Obrázok 3.2: Ukážka textovej definície

### 3.2.1 Spracovanie textovej definície a generovanie dát

Textová definícia je spracovaná po častiach. Najprv sú spracované príčiny a dôsledky do zoznamov c*auses* a e*ffects*. V týchto zoznamoch sú uložené identifikátory a popisy jednotlivých príčin a dôsledkov. Následne sú spracované pravidlá. Pokiaľ sa na ľavej strane pravidla vyskytne príčina, alebo už predtým definovaný medziuzol, tak spracovanie skončí so sémantickou chybou. Ak sa počas spracovania vyskytne na pravej strane pravidla dôsledok, alebo nedefinovaný identifikátor, tak nastane sémantická chyba. Ak sa na pravej strane pravidla vyskytne viac typov operátorov, tak spracovanie tiež končí so sémantickou chybou.

Ak je na ľavej strane pravidla nedefinovaný identifikátor, tak je vytvorený medziuzol s daným identifikátorom a zaradený do zoznamu medziuzlov. Tomuto novovytvorenému uzlu je priradený typ na základe operátoru v pravidle a zoznam odkazov na už definované uzly na pravej strane pravidla. Pre prípad `int1 = p1 || p2` by bol vytvorený medziuzol `int1` s priradeným typom OR a zoznamom odkazov na príčiny `p1` a `p2`. Ak je na ľavej strane dôsledok, tak sú tieto dáta priradené odpovedajúcemu dôsledku v zozname *effects*.

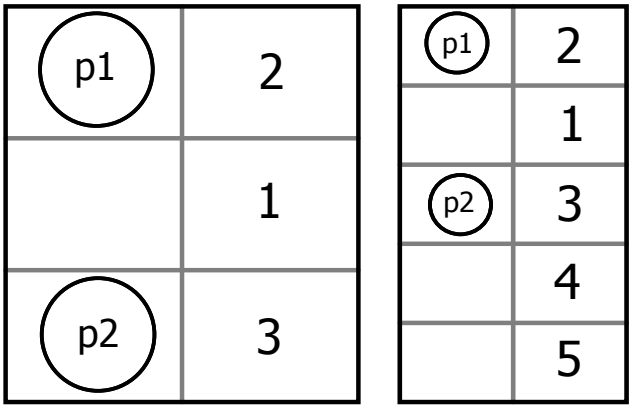
Obmedzenia sú spracované obdobne. Ak je v definícií obmedzenia nedefinovaný identifikátor, spracovanie končí chybou, inak je vytvorené nové obmedzenie s priradeným typom obmedzenia na základe písmena pred znakom ':'. Ak ide o obmedzenie *exclusive, inclusive*, alebo *one*, tak sa ku novému obmedzeniu pridáva zoznam odkazov na uzly týkajúcich sa tohto obmedzenia. V prípade, že ide o obmedzenie *requires*, alebo *masks*, tak je s novým obmedzením uložený odkaz na ľavý a pravý uzol v obmedzení.

Súčasne s typom a zoznamom predchádzajúcich uzlov je uzlom priraďované číslo stĺpca pre určenie x-súradnice pri vykresľovaní grafu. Príčiny majú implicitne priradené číslo stĺpca 0. Medziuzly majú túto hodnotu určenú podľa predchádzajúcich uzlov. Pri vytváraní nového uzla je vybraná najvyššia hodnota stĺpca medzi predchodcami a novému uzlu priradená hodnota o 1 vyššia. Všetky dôsledky majú rovnaké číslo stĺpca a to o 1 vyššiu ako medziuzol s najväčším číslom. Obmedzenia nemajú údaj o hodnote stĺpca.

#### 3.2.1.1 Mriežka nad uzlami

Pri určovaní y-súradnice medziuzlov by sa dalo postupovať tak, že sa vypočíta ako priemer y-súradníc predchodcov. Toto určenie nestačí, pretože dva rôzne medziuzly s rovnakými predchodcami by mali rovnakú polohu a tým pádom by sa prekrývali. Preto je potrebné pri vytváraní nového uzlu zistiť obsadenosť vypočítaného miesta a v prípade, že je toto miesto obsadené, tak určiť iné miesto.

Na obrázku 3.3 je vidieť mriežka vytvorená nad dvoma príčinami. Počet riadkov mriežky závisí od maximálneho počtu uzlov v stĺpci. Počet stĺpcov mriežky závisí od najväčšieho čísla stĺpca medzi uzlami. Bunka v mriežke obsahuje informáciu o obsadenosti bunky a y-súradnicu prislúchajúcu bunke. Prípad vľavo na obrázku 3.3 má tri riadky. Mriežka má vždy počet riadkov rovný dvojnásobku maximálneho počtu uzlov v stĺpci mínus jeden. Ak by mal byť vytvorený nový uzol s predchodcami `p1` a `p2`, tak výsledná y-súradnica je najbližšie bunke označenej číslom 1, ak by bolo toto miesto obsadené iným uzlom, tak by bola zvolená bunka s číslom 2. Číslo v bunke určuje prioritu pri určovaní miesta medziuzlu s predchodcami `p1` a `p2`. Pokiaľ nie je nájdené miesto pre nový uzol, tak je mriežka rozšírená pre ďalšie voľné bunky ako na obrázku 3.3 vpravo. Určovanie polohy podľa mriežky sa týka len medziuzlov. Dôsledky sú vykresľované nezávisle od ostatných uzlov.




Obrázok 3.3: Na tomto obrázku je znázornená priorita miest na ktoré bude pridávaný medziuzol s predchodcami `p1` a `p2`. Ak by mal byť vytvorený nový medziuzol s predchodcami `p1` a `p2`, tak bude pridaný na miesto označené ako 1. Ak by bolo toto miesto obsadené, tak by bol pridaný na miesto 2. V prípade obsadenosti miest 1 a 2 by sa algoritmus pokúsil obsadiť miesto 3. V mriežke na ľavej strane obrázku môže nastať situácia kedy by boli obsadené miesta 1, 2 a 3 a užívateľ bude chcieť pridať medziuzol nad `p1` a `p2`. V tom prípade sa tabuľka rozšíri na tabuľku na pravej strane obrázku a nový uzol sa pridá na miesto označené ako 4.

## 3.3 Reprezentácia grafu

Dáta, ktoré sú ukladané o uzloch a obmedzeniach sa trochu líšia podľa toho či sa jedná o príčinu, dôsledok, medziuzol, alebo obmedzenie. Položka `col` obsahuje číslo stĺpca daného uzlu, `formula` obsahuje informácie pre automatické generovanie testovacích prípadov v rozhodovacej tabuľke. Každý uzol obsahuje položku `name` udávajúcu identifikátor uzla. Príčiny a dôsledky obsahujú atribút `value`, ktorá udáva popis uzlu z textovej definície. Každý uzol a obmedzenie obsahuje položku `type` určujúcu či sa jedná o príčinu, dôsledok, medziuzol, alebo obmedzenie. Typ obmedzenia je určený atribútom `node`. Medziuzly a dôsledky obsahujú atribút `op`, ktorý určuje typ

uzlu. Zoznam odkazov na predchádzajúce uzly medziuzlov a dôsledkov je uložený v položke `left`. Obmedzenia typu *exclusive, inclusive* a *one* majú položku `list`, ktorá obsahuje zoznam príčin týkajúcich sa daného obmedzenia. Obmedzenia *requires* a *masks* majú atribút `left`, obsahujúci odkaz na ľavý uzol v obmedzení a `right`, obsahujúci odkaz na pravý uzol v obmedzení. Pre obmedzenie `p1->p2` by sa vytvorilo obmedzenie s nasledujúcimi atribútmi: `type` s hodnotou "restriction", `node` s hodnotou "R", `left` s odkazom na príčinu `p1`, `right` s odkazom na príčinu `p2`. Atribút `state` je využívaný pri vyhodnocovaní dôsledkov v rozhodovacej tabuľke. Položky `x` a `y` majú hodnotu udávajúcu x-súradnicu a y-súradnicu. Atribút `r` udáva polomer uzlu, alebo obmezenia vykresleného v grafe. Hodnoty `x`, `y` a `r` sú vypočítané na základe veľkosti elementu v prehliadači v ktorom sa graf nachádza, čísla stĺpca a voľného miesta v mriežke uzlov.

```
col:0
formula: "p1"
name: "p1"
r: 20
state: 0
type: "cause"
value: "text"
x: 180
y: 60
```

```
col:1
formula:
Logic.and(p1,p2)
left: [p1,p2]
name:"int1"
op: "&&"
r: 20
state: false
type: "explicit"
x: 540
y: 122
```

```
list:[p1,p2]
node: "E"
r: 10
type:
"restriction"
x: 90
y: 122
```

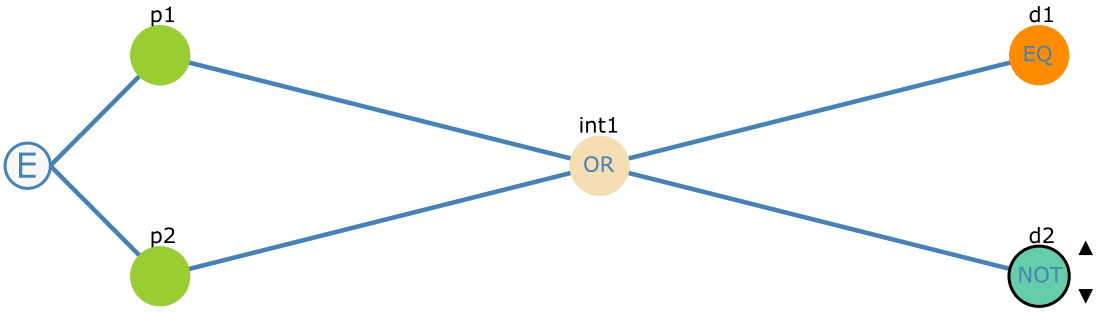
```
left: p1
node: "R"
r: 10
right: p2
type:
"restriction"
x: 820
y: 180
```

Obrázok 3.4: Príklady objektov reprezentujúcich uzly a obmedzenia. V prvej časti je príklad objektu, ktorý reprezentuje príčinu s identifikátorom `p1` a popisom `text`. V druhej časti je objekt medziuzlu s identifikátorom `int1` typu AND a predchodcami `p1` a `p2` a číslom stĺpca 1. V tretej časti je obmedzenie *exclusive* nad príčinami `p1` a `p2`. Vo štvrtej časti je *requires* obmedzenie `p1->p2`.

# 3.4 Vykreslený graf

Podľa dát popísaných v kapitole 3.3 sú vykresľované jednotlivé uzly a obmedzenia do grafu. Uzly sú vykreslené s definovaným identifikátorom a typom uzlu. Obmedzenia sú vykreslené ako uzly s typom obmedzenia namiesto typu uzlu. Každý vykreslený objekt v grafe reaguje na kliknutie.

Po kliknutí na uzol je tento uzol braný ako označený a je odlíšený od neoznačených uzlov inou farbou. Ak užívateľ klikne na hranu medzi uzlami, tak sú označené uzly, medzi ktorými je vedená táto hrana. Po kliknutí na obmedzenie je označené obmedzenie a uzly týkajúce sa obmedzenia. Ak je označená jedna príčina, alebo jeden dôsledok, tak sú vedľa uzlu vykreslené objekty, ktoré slúžia na posun uzlov nahor a nadol. Na obrázku 3.5 je vykreslený graf s dvoma príčinami `p1` a `p2`, s jedným medziuzlom `int1` typu OR s predchodcami `p1` a `p2`, dvoma dôsledkami `d1` a `d2`, jedným obmedzením *exclusive* nad uzlami `p1` a `p2`, označeným dôsledkom `d2` a objektmi pre posúvanie uzlov napravo od dôsledku `d2`.




Obrázok 3.5: Vykreslený CEG graf

### 3.4.1 Interaktívne zmeny v grafe

Od nástroja sa očakáva, že bude užívateľovi umožnené vykonávať zmeny v grafe interaktívne pomocou grafu, to znamená bez editácie textovej definície. Okolnosti, za akých sú zmeny vykonávané, sú popísané v kapitole 3.6. Vzťah medzi zmenami v grafe a textovou definíciou je popísaný v kapitole 3.5.

Ak užívateľ pridá uzol určitého typu, tak je vytvorený nový uzol s daným typom a generickým identifikátorom. Vytvorený identifikátor je číslo, ktoré je prednastavené na 1. V prípade, že sa v dátach vyskytujú iné číselné identifikátory, tak je z týchto čísel vybrané maximum a vytvorí sa identifikátor s číslom o 1 vyššie ako maximum. Tento uzol sa následne vloží medzi dáta popísané v kapitole 3.3. Obmedzenia sú pridané obdobne, len sú uložené bez identifikátorov.

Keď je uzol zmazaný užívateľom, tak je odobraný z dát. Následne musia byť odobraté aj všetky výskyty odobraného uzlu v pravidlách a obmedzeniach. To znamená, že sú zmazané výskyty odobraného uzla v atribúte `left` vo všetkých uzloch a v atribútoch `list`, `left` a `right` v obmedzeniach. Obdobne sú mazané aj obmedzenia.

Ak sa pridáva hrana medzi dvoma uzlami, tak atribútu `left` uzlu s väčším číslom stĺpca je pridaný odkaz na uzol s menším číslom stĺpca. Mazanie hrany prebieha obdobne. Zo zoznamu `left` v uzle s vyšším číslo stĺpca je odstránený odkaz na uzol s menším číslom stĺpca.

Pri editácií je len zmenená hodnota atribútu `op` meneného uzla.

Keďže príčiny a dôsledky sú vykresľované na základe poradia v definícií a dátach. Tak pre posun uzla stačí zmeniť poradie v dátach, následne zmeniť poradie v textovej definícií a graf prekresliť s novým poradím.

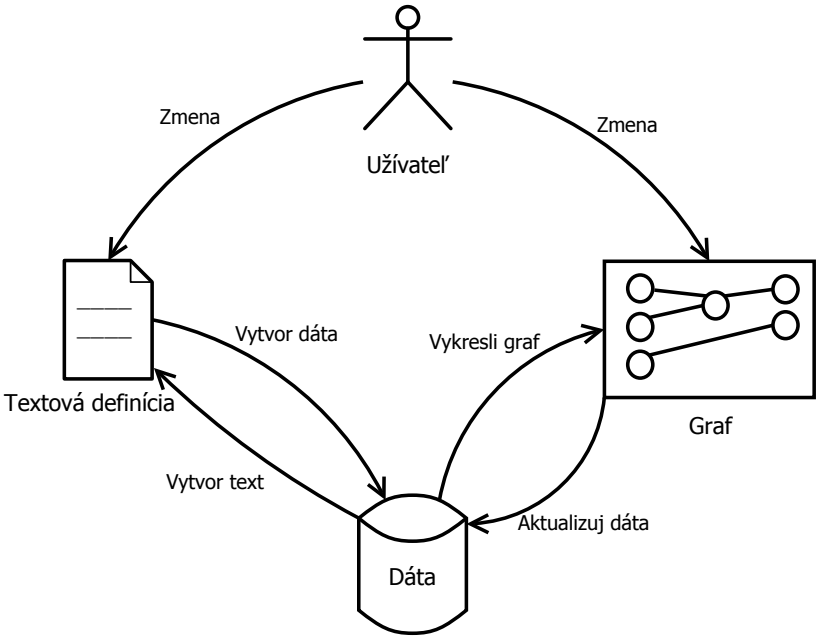
## 3.5 Princíp fungovania

Na obrázku 3.6 je znázornené správanie nástroja. *Textová definícia* reprezentuje textový dokument s textovou definíciou grafu. *Graf* predstavuje element v prehliadači s vykreslenými objektmi. *Dáta* predstavujú uložené informácie o každom uzle a obmedzení, ktoré sú prítomné v grafe. Užívateľ môže meniť textovú definíciu a vykonávať zmeny v grafe. Ak užívateľ zmení textovú definíciu a je bez syntaktickej, alebo sémantickej chyby, tak sú z nej vytvorené dáta. Následne je na základe týchto dát vykreslený graf. Zmena v grafe vykonaná užívateľom spôsobí, že sa aktualizujú uložené dáta o uzloch a následne sú tieto dáta transformované do textu a vložené do

textovej definície. Následné spracovanie textovej definície spôsobí, že sú prepočítané polohy a rozmery uzlov a graf sa prekreslí s novými dátami. Pri niektorých operáciách s grafom je prekreslenie grafu nežiadúce a po zmene grafu a následnej zmene textovej definície nedôjde ku spracovaniu textovej definície. Sú to operácie:

- Zmazanie uzlu, obmedzenia, alebo hrany.
- Pridanie hrany, alebo obmedzenia
- Zmena typu uzlu

Ostatné operácie s grafom prekreslia graf s inými polohami a rozmermi uzlov, ak boli operáciou zmenené.




Obrázok 3.6: Princíp fungovania nástroja *ceg-editor*.

# 3.6 Správanie tlačidiel

Každé tlačidlo vykonáva inú akciu. Pri niektorých sa líši akcia a aktívnosť od stavu v akom sa graf nachádza. Tlačidlá "Export", "Export_dropdown" "Import", "+Cause", "+Effect" sú aktívne za každých okolností a žiadny stav grafu sa na ne nevzťahuje. V tabuľke 3.1 v druhom stĺpci sú popísané akcie, ktoré sa vykonávajú po stlačení daného tlačidla. Okolnosti za akých sú tlačidlá aktívne sú popísané v tabuľke v treťom stĺpci. Okolnosti tlačidla "Undo" sú popísané v kapitole 3.6.1. Tlačidlá sú vymenovávané v poradí zľava doprava z obrázku 3.7.


Obrázok 3.7: Tlačidlá v hornej časti nástroja.

| Tlačidlo | Akcia | Okolnosti |
|---|---|---|
| "Export" | Exportuje celý graf vrátane textovej definície a testovacích prípadov vo formáte JSON. | Vždy |
| "Export" | Zobrazí sa menu s nasledujúcimi možnosťami, | Vždy |

| | | |
|---|---|---|
| dropdown | po stlačení jednej z možností je vykonaná akcia popísaná v zátvorke:<br>• *Test Cases* (uloženie testovacích prípadov vo formáte CSV)<br>• *PNG* (uloženie grafu vo formáte PNG)<br>• *SVG* (uloženie grafu vo formáte SVG)<br>• *Report* (uloženie reportu vo formáte Markdown)<br>• *PDF* (uloženie reportu vo formáte PDF) | |
| "Import" | Importovanie grafu zo súboru do nástroja. | Vždy |
| "Undo" | Ak je tlačidlo v stave "Undo", stav grafu sa vráti o krok späť. Ak je tlačidlo v stave "Redo", vrátená akcia je vykonaná opäť. | Popísané v kapitole 3.6.1 |
| "+Cause" | Pridaná príčina s generickým identifikátorom. | Vždy |
| "Cause" dropdown | Zobrazí sa menu s nasledujúcimi možnosťami, po stlačení jednej z možností je vykonaná akcia popísaná v zátvorke:<br>• *Exclusive* (pridané obmedzenie *exclusive* nad označenými uzlami)<br>• *Inclusive* (pridané obmedzenie *inclusive* nad označenými uzlami)<br>• *One* (pridané obmedzenie *one* nad označenými uzlami)<br>• *Requires* (pridané obmedzenie *requires* nad označenými uzlami) | Ak sú označené minimálne dve príčiny. Pre položku *Requires* v menu platí, že je platná iba ak sú označené presne dve príčiny. Ostatné položky sú platné ak sú označené minimálne dve príčiny. |
| "+Effect" | Pridaný dôsledok s generickým identifikátorom. | Vždy |
| "Effect" dropdown | Zobrazí sa menu s nasledujúcimi možnosťami, po stlačení jednej z možností je vykonaná akcia popísaná v zátvorke:<br>• *Masks* (pridané obmedzenie *masks* nad označenými uzlami) | Ak sú označené 2 dôsledky. |
| "+AND" | Pridaný medziuzol, ktorý je typu *AND* s označenými uzlami ako predchodcami. | Sú označené minimálne dva uzly, medzi ktorými nie je dôsledok. |
| "+OR" | Pridaný medziuzol, ktorý je typu *OR* s označenými uzlami ako predchodcami. | Sú označené minimálne dva uzly, medzi ktorými nie je dôsledok. |
| "+NOT" | Pridaný medziuzol, ktorý je typu *NOT* s označeným uzlom ako predchodcom. | Je označený jeden uzol, ktorý nie je dôsledok |
| "+Edge" | Pridaná hrana medzi označenými uzlami | Sú označené dva uzly a nie je medzi nimi prítomná hrana. Uzly majú rôzne čísla stĺpcov. |
| "Edit" | Zobrazí sa menu s nasledujúcimi možnosťami, po stlačení jednej z možností je vykonaná | Je označený jeden uzol, ktorý nie je príčina. |

| | | |
|---|---|---|
| | akcia popísaná v zátvorke:<br>• *AND* (typ označeného uzla je zmenený na *AND*)<br>• *OR* (typ označeného uzla je zmenený na *OR*)<br>• *NOT* (typ označeného uzla je zmenený na *NOT*)<br>• *EQ* (typ označeného uzla je zmenený na *EQ*) | Položky *NOT* a *EQ* sú aktívne pokiaľ označený uzol má maximálne jedného predchodcu (jeden odkaz v `left`). Položky *AND* a *OR* sú aktívne pokiaľ je aktívne tlačidlo "Edit" |
| "Unselect" | Zrušenie označenia všetkých označených uzlov | Ak je označený minimálne jeden uzol |
| "Delete" | Ak je označený jeden uzol, tak je daný uzol zmazaný. Ak sú označené dva uzly a je medzi nimi prítomná hrana, tak je zmazaná hrana medzi týmito uzlami. | Ak je označený jeden uzol, alebo dva uzly medzi ktorými je prítomná hrana. |

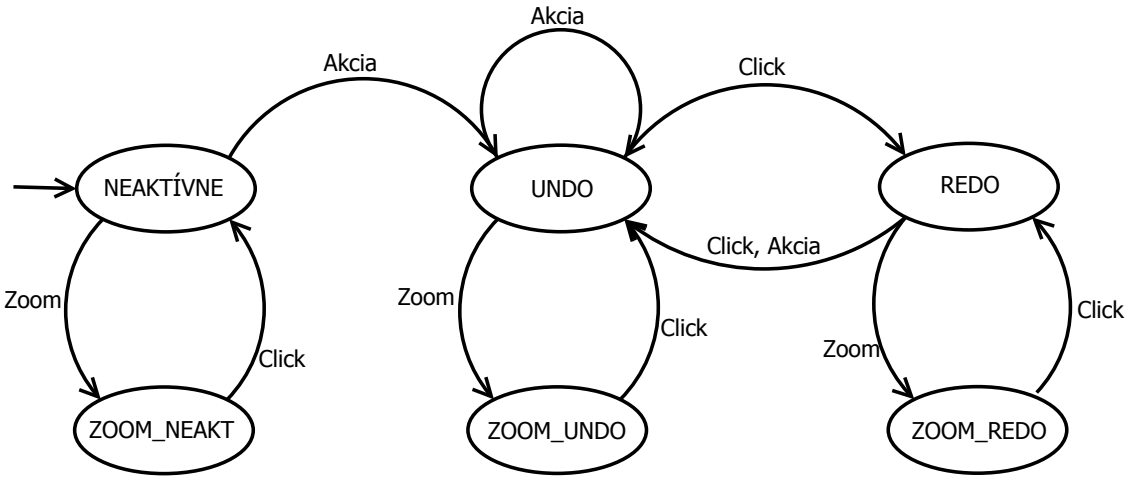
Tabuľka 3.1: Akcie jednotlivých tlačidiel

### 3.6.1 Tlačidlo "Undo"

Funkcionalita tohto tlačidla sa mení podľa stavu v akom sa aplikácia nachádza. Okolnosti za akých sa "Undo" mení sú znázornené na obrázku 3.8. *Akcia* predstavuje udalosť, ktorá je schopná zmeniť stav tlačidla. Patrí sem:

- pridanie uzlu, hrany, alebo obmedzenia
- zmazanie uzlu, hrany, alebo obmedzenia
- presun príčiny, alebo dôsledku
- zmena typu uzlu

*Click* predstavuje kliknutie na tlačidlo a *Zoom* znamená priblíženie v grafe. Po spustení nástroja je tlačidlo neaktívne, následne, ak je vykonaná *akcia*, tak sa stane tlačidlo aktívnym. Po stlačení tlačidla v tomto stave je vrátená predošlá akcia a tlačidlo prejde do stavu *redo*. Ak v tomto stave užívateľ stlačí tlačidlo, tak je vrátená akcia vykonaná znova a tlačidlo prejde do stavu *undo*. Ak v akomkoľvek stave užívateľ priblíži graf, teda vykoná *zoom*, tak tlačidlo prejde do stavu, v ktorom pokiaľ ho užívateľ stlačí, tak je priblíženie grafu zrušené a tlačidlo sa vráti do stavu, ktorom bol vykonaný *zoom*.




Obrázok 3.8: Prechody tlačidla "Undo".

# 3.7 Rozhodovacia tabuľka

Požiadavky na rozhodovaciu tabuľku sú:

- Uvedenie všetkých príčin a dôsledkov prítomných v grafe s ich popisom.
- Pridávanie testovacích prípadov do tabuľky.
- Zadávanie hodnôt príčin do tabuľky a výpočet hodnoty dôsledkov pre uvedenú kombináciu.
- Informácia užívateľovi o pokrytí príčin a dôsledkov.
- Zmena popisu príčiny, alebo dôsledku z rozhodovacej tabuľky.
- Mazanie pridaných stĺpcov v tabuľke.
- Označenie testovacieho prípadu, ktorý nevyhovuje obmedzeniu.

V tabuľke sú najprv uvedené príčiny a po nich všetky dôsledky. Aby bola splnená požiadavka na uvedenie všetkých príčin a dôsledkov prítomných v grafe, tak pri každom pridaní a odobratí uzlu je tabuľka aktualizovaná. Ak sa nachádza v tabuľke identifikátor, ktorý je nedefinovaný, tak je tento riadok odstránený. Ak je medzi príčinami, alebo dôsledkami identifikátor neprítomný v tabuľke, tak je riadok s vloženými hodnotami identifikátora a popisu príčiny, alebo dôsledku pridaný do tabuľky.

Po stlačení tlačidla "+" (na obrázku 3.9 vpravo hore) je pridaný nový stĺpec do tabuľky. Novovytvorená bunka v hlavičke má vygenerovanú hodnotu na základe počtu pridaných stĺpcov a je editovateľná. Ak sa nachádza kurzor v jednej z novovytvorených buniek, tak sa zobrazí tlačidlo "-". Po kliknutí naň je stĺpec zmazaný. Nové bunky v tabuľke v riadkoch s príčinami sú editovateľné a užívateľ môže do nich zadávať hodnoty. Tieto hodnoty sú interpretované buď ako pravda, nepravda, alebo chybná hodnota. Vstupy interpretované ako pravda sú: "1", "t", "T", "true", "TRUE", "yes", "YES", "y", "Y". Vstupy interpretované ako nepravda sú: "0", "f", "F", "false", "FALSE", "no", "NO", "n", "N". Ostatné hodnoty sú interpretované ako chybná hodnota. Ak je zadaná chybná hodnota, bunka v ktorej sa hodnota nachádza je podfarbená na červeno.

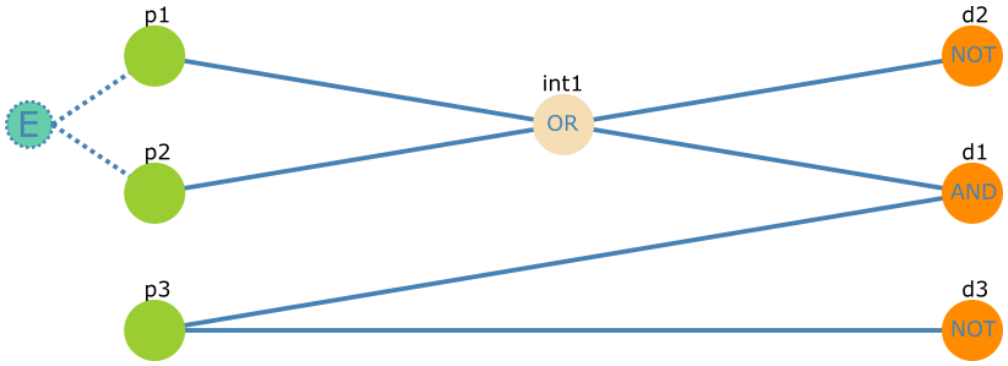
Po zadaní hodnôt príčin v ľubovoľnom stĺpci sú vyhodnotené dôsledky a výsledné hodnoty vložené do tabuľky. Výsledný stĺpec reprezentuje jeden testovací prípad. Vyhodnocovaniu dôsledkov sa venuje kapitola 3.7.1.

Informovanie užívateľa o pokrytí príčin a dôsledkov je dosiahnuté podfarbením riadkov. Ak sa príčina, alebo dôsledok nachádza v oboch stavoch (platný, neplatný), tak je riadok prislúchajúci danej príčine, alebo dôsledku podfarbený na modro. Ak sa nachádza len v jednom z možných stavov, alebo sa v riadku vyskytuje nevyplnená bunka, tak je riadok podfarbený na oranžovo. Na obrázku 3.9 sú príčiny `p2` a `p3` a dôsledky `d1` a `d3` pokryté v oboch stavoch. Príčina `p1` a dôsledok `d2` sú pokryté len v jednom zo stavov, preto sú podfarbené na oranžovo.

Súčasťou požiadaviek je aj hlásenie o testovacom prípade, ktorý nevyhovuje obmedzeniam. Pri vyhodnocovaní tabuľky je kontrolované, či daná kombinácia príčin vyhovuje všetkým obmedzeniam. V prípade na obrázku 3.9 nevyhovuje testovací prípad `[1]` obmedzeniu *exclusive* nad uzlami `p1` a `p2`. Po kliknutí na tlačidlo "Show", sú v grafe vyznačené nevyhovujúce obmedzenia viz obrázok 3.10.

| Name | Description | [1] Show | [2] | + |
|---|---|---|---|---|
| p1 | The first character is 'A' | 1 | 1 | |
| p2 | The first character is 'B' | 1 | 0 | |
| p3 | The second character is a digit | 0 | 1 | |
| d2 | Exception X12 | false | false | |
| d1 | Update of a file | false | true | |
| d3 | Exception X13 | true | false | |

Obrázok 3.9: Ukážka vyplnenej rozhodovacej tabuľky s nevyhovujúcim obmedzením




Obrázok 3.10: Graf so zvýraznenou nevyhovujúcou podmienkou po stlačení tlačidla "Show"

Obmedzenie *masks* spôsobuje to, že na základe hodnoty jedného dôsledku mení hodnotu druhého dôsledku na *false*. Toto obmedzenie sa aplikuje po vyhodnotení všetkých dôsledkov dodatočne.

### 3.7.1 Vyhodnocovanie grafu

Po vyplnení hodnôt príčin v stĺpci sú vyhodnotené dôsledky pre danú kombináciu príčin. Toto vyhodnocovanie prebieha zľava doprava. Najprv sú nastavené atribúty `state` všetkým príčinám na hodnoty z tabuľky. Následne sú postupne vyhodnocované medziuzly. Ak je napríklad medziuzol definovaný ako `int1 = p1 || p2` a `p1` má hodnotu `state` rovnú 0 a `p2` má hodnotu `state` rovnú 1, tak sa uzlu `int1` nastaví atribút `state` na 1. Týmto spôsobom sú vyhodnotené všetky uzly v grafe a výsledne hodnoty dôsledkov sú vložené do tabuľky do príslušného stĺpca.

### 3.7.2 Automatické generovanie testovacích prípadov

Zatiaľ čo pre tvorbu testovacích prípadov po vyplnení príčin stačilo vyhodnocovanie grafu zľava doprava, pre automatické generovanie testovacích prípadov bude potreba vyhodnocovať graf aj sprava doľava. Bude potrebné zistiť, za akých okolností je daný dôsledok platný a za akých okolností platný nie je. Tento problém sa v informačných technológiách nazýva ako *boolean satisfiability problem* (skrátene *SAT*) a zaoberá sa hľadaním okolností za akých je daná logická formula platná. Skúma, aké premenné je potrebné nastaviť na platnú a neplatnú hodnotu, aby formula platila. S využitím nástroja, ktorý by dokázal tento problém riešiť [11], by sa nástroj *ceg-editor* mohol dotazovať na okolnosti, za akých je určitý dôsledok platný, alebo neplatný.

Pojmy *combinatorial coverage* (pokrytie všetkých kombinácií hodnôt klauzúl), *predicate coverage* (pokrytie všetkých predikátov) a *clause coverage* (pokrytie všetkých klauzúl) [5] sú pojmy týkajúce sa pokrytia kódu v testovaní softwaru. Predikát je logický výraz, ktorý sa vyhodnocuje na pravdivú, alebo nepravdivú hodnotu. Obsahuje logické spojky a premenné. Klauzula je logický výraz bez logických spojok. Napríklad majme výraz `a OR b`. Celý výraz je predikát a `a` a `b` sú klauzule. Na to aby bolo dosiahnuté pokrytie klauzúl, musia byť `a` aj `b` vyhodnotené na true aj false. Na to aby bolo dosiahnuté pokrytie predikátov, tak sa musí celý výraz `a OR b` nachádzať v stave true aj false. V nástroji *ceg-editor* predstavujú príčiny klauzule a dôsledky predikáty.

Užívateľ má v nástroji *ceg-editor* na výber 3 typy pokrytia príčin a dôsledkov.

- *CoC* - c*ombinatorial coverage* (pokrytie všetkých kombinácií hodnôt klauzúl) - sú vyhodnotené všetky kombinácie príčin.
- *PP* - *predicate coverage* (pokrytie všetkých predikátov) - sú pokryté všetky dôsledky (nachádzajú sa aj v stave platný aj neplatný).
- *CC* - c*lause coverage* (pokrytie všetkých klauzul) - sú pokryté všetky príčiny.

Po stlačení tlačidla "CoC", "PP", alebo "CC" sú vygenerované testovacie prípady s odpovedajúcim pokrytím. Podrobnosti implementácie generovania testovacích prípadov sú v kapitole 4.2.

## 3.8 Výstupy nástroja *ceg-editor*

Výstupom nástroja *ceg-editor* môžu byť:

- Testovacie prípady.
- Samotný graf .
- Celkový report s testovacími prípadmi, textovou definíciou a grafom.
- Textová definícia, testovacie prípady a graf pre import.

Testovacie prípady sú ukladané vo formáte *csv*. Ide o rozhodovaciu tabuľku uloženú v *csv*. Stĺpce sú oddelené čiarkou a riadky sú oddelené znakom konca riadku. Ukážka uložených testovacích prípadov je na obrázku 3.11. Graf je ukladaný vo formáte SVG. Celkový report je uložený vo formáte

*Markdown* a je uvedený v prílohe B. Celý graf pre import je uložený vo formáte JSON, schéma JSONu [9] je uvedená na obrázku 3.12.

```
Name,Description,[1],[2],[3],[4],[5]
1,The first character is 'A',0,0,1,1,0
2,The first character is 'B',0,1,0,0,0
3,The second character is a digit,1,1,1,0,0
71,Exception X12,true,false,false,false,true
70,Update of a file,false,true,true,false,false
72,Exception X13,false,false,false,true,false
```

Obrázok 3.11: Príklad uloženia testovacích prípadov vo formáte *csv*

```
{
    "title": "Graph to export",
    "type": "object",
    "properties": {
        "projectName": {
            "type": "string"
        },
        "textDef": {
            "type": "string"
        },
        "testCases": {
            "type": "string",
        }
    },
}
```

Obrázok 3.12: Schéma JSON exportovaného grafu

# 4 Popis implementácie nástroja ceg-editor

V tejto časti je čitateľ oboznámený s použitými technológiami a ich stručným popisom v kapitole 4.1. V kapitole 4.2 je priblížená implementácia automatického generovanie testovacích prípadov. V kapitole 4.3 sú zhrnuté nedostatky nástroja a návrh ich riešení.

## 4.1 Popis technológií týkajúcich sa nástroja ceg-editor

Medzi použité technológie v nástroji *ceg-editor* patrí HTML, CSS, Javascript, Markdown, SVG a D3.js. Patrí sem aj nástroj *logic-solver* [11], ktorého funkcionalita je popísaná v kapitole 4.2.

### 4.1.1 HTML, CSS, Javascript

HTML (Hypertext Markup Language) je značkovací jazyk používaný na vytváranie webových stránok. CSS (Cascading Style Sheets) udáva ako sú HTML elementy zobrazené v prehliadači. Javascript je interpretovaný jazyk najčastejšie používaný ako skriptovací jazyk pre webové stránky.

*Browserify* [13] je nástroj, ktorý umožňuje použiť metódu *require* vo webových prehliadačoch. *Require* slúži na dynamické načítavanie modulov. Nástroj *ceg-editor* využíva *browserify* na umožnenie načítania nástroja *logic-solver*.

### 4.1.2 SVG

SVG (Scalable Vector Graphics) [8] je značkovací jazyk pre popis dvojrozmerných objektov pomocou XML. SVG umožňuje definíciu troch typov objektov:

- vektorové grafické objekty
- obrázky (rastrové)
- text

Objekty definované SVG môžu byť ľubovoľne zoskupované, štylizované a transformované. V nástroji *ceg-editor* je SVG využité na zobrazovanie CEG grafu nástrojom D3.js.

### 4.1.3 Markdown

Markdown [7] je značkovací jazyk vytvorený za účelom konverzie do HTML. Hlavnou myšlienkou tohto formátu je, aby bol publikovateľný a čítateľný aj bez konverzie. Má jednoduchú syntax. Je používaný pre readme súbory, alebo weblogy. Pomocou jednoduchých značiek je možné vkladať nadpisy, zoznamy, alebo obrázky. V nástroji *ceg-editor* je tento jazyk použitý ako report s testovacími prípadmi.

V tabuľke 4.1 je príklad prekladu jazyka Markdown do HTML. V ľavej časti tabuľky je text vo formáte Markdown a v pravej časti je ukážka, ako by približne vyzeral text preložený do HTML v prehliadači.

| | |
|---|---|
| `# Nadpis`<br>`## Podnadpis`<br>`### Ďalší podnadpis`<br>`Atribúty textu _italic_, __bold__`<br>`Odrážky:`<br>`* odrážka`<br>`* odrážka`<br>`* odrážka` | **Hlavný nadpis**<br>**Podnadpis**<br>**Ďalší podnadpis**<br>Atribúty textu ***italic***, bold<br>Odrážky:<br>• odrážka<br>• odrážka<br>• odrážka |

Tabuľka 4.1: Vzor značkovacieho jazyka Markdown

### 4.1.4 PEG.js

Nástroj PEG.js [10] je generátor parseru pre Javascript. Tento nástroj po zadaní formálnej gramatiky vygeneruje parser, ktorý je schopný spracovať jazyk popísaný zadanou gramatikou. Dokáže informovať o syntaktických chybách, ktoré v priebehu spracovania nastali. V nástroji *ceg-editor* bol použitý na vygenerovanie parseru, ktorý spracováva textovú definíciu CEG.

### 4.1.5 D3.js

D3.js [12] je Javascriptová knižnica, ktorá umožňuje vizualizáciu dát vo webových prehliadačoch. Využíva technológie HTML, CSS a SVG. Umožňuje vybrať uzol z Document Object Modelu a manipulovať s ním. V nástroji *ceg-editor* je použitý na vytvorenie SVG elementov reprezentujúcich uzly a hrany CEG s danými atribútmi a akciami.

## 4.2 Implementácia generovania testovacích prípadov

Generovanie testovacích prípadov je implementované s pomocou nástroja *logic-solver* [11]. V nasledujúcom príklade je možné vidieť príklad použitia tohto nástroja. V prvých dvoch riadkoch príkladu sú nástroju predložené obmedzenia:

- nech A nie je súčasne platné s B.
- nech je platné B, alebo C.

```
solver.require(Logic.atMostOne("A", "B"));
solver.require(Logic.or("B", "C"));
solver.solveAssuming("A")
```

Posledný príkaz žiada od nástroja takú kombináciu premenných A, B a C, pri ktorých bude vyhovené obmedzeniam a súčasne bude platné A. Výsledok dotazu je, že na to aby bolo vyhovené predtým stanoveným obmedzeniam, musí byť platné A a súčasne C.

Výsledok dotazu je vždy iba jeden. Na to, aby bolo možné vymenovať všetky riešenia, je potrebná metóda `forbid(curSol.getFormula())`, ktorej je predané súčasné riešenie ako parameter. Táto metóda spôsobí, že následné dotazy nebudú uvádzať predchádzajúce riešenie. Riešenie automatického generovania testovacích prípadov je obdobné. Nástroju *logic-solver* je predaná definícia grafu vrátane obmedzení a následne je možné sa dotazovať na platné riešenia grafu.

Pri generovaní prípadov s kombinačným pokrytím sa dotazuje na všetky riešenia vyhovujúce obmedzeniam v grafe. Pri pokrytí dôsledkov sú prechádzané všetky dôsledky a najprv sa dotazuje na riešenie grafu, kedy je daný dôsledok platný a následne sa dotazuje na riešenie kedy je daný dôsledok neplatný. Takýmto spôsobom je dosiahnuté pokrytie dôsledkov. Podobne je dosiahnuté aj pokrytie príčin, len sa prechádzajú všetky príčiny.

## 4.3 Nedostatky nástroja ceg-editor a návrh ich riešení

Nástroj *ceg-editor* má pár nedostatkov, ktoré by bolo vhodné počas budúcej práce vyriešiť.

### 4.3.1 Umožnenie tvorby rovnakých uzlov

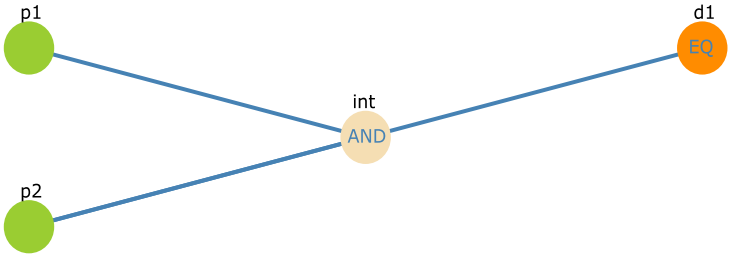
Pri zadávaní textovej definície, je možné vytvoriť dva uzly, ktoré majú rovnakých predchodcov aj typ. Riešenie je rozšírenie kontroly predchádzajúcej definície o typ a zoznam predchodcov.

### 4.3.2 Obmedzenie *masks*

Obmedzenie *masks* mení hodnotu dôsledku na základe hodnoty iného dôsledku. Toto obmedzenie je aplikované až po vyhodnotení všetkých dôsledkov Pri automatickom generovaní testovacích prípadov môže nastať situácia kedy by sa nástroj dotazoval na platný dôsledok, no vo výsledku by sa vyskytoval ako neplatný. Riešením je použitie dočasného uzlu

### 4.3.3 Prekrývanie hrán

V grafe môže nastať situácia, kedy sa hrany navzájom prekrývajú. V prípade na obrázku 4.1 to je hrana medzi `p2` a `d1` a hrana `p2` a `int`. Riešením je zavedenie z-súradníc nad uzlami a v prípade prechodu cudzej hrany cez uzol s inou súradnicou by bol tento prechod odlíšený.




Obrázok 4.1: Prekrývanie hrán v grafe

### 4.3.4 Prekrývanie obmedzení

Pri vytváraní obmedzení môže nastať situácia, kedy sa po vykreslení v grafe obmedzenia prekrývajú, riešením je vytvorenie mriežky nad obmedzeniami.

# 5 Záver

Cieľom práce bolo vytvoriť nástroj na tvorbu a editáciu grafov príčin a dôsledkov. Hlavnou motiváciou pre tvorbu nástroja bolo, že v súčasnosti existuje len jeden nástroj využívajúci metódu CEG, ide však o proprietárny software. Nástroj *ceg-editor* podporuje textovú definíciu príčin, dôsledkov a obmedzení v jazyku vytvorenom pre CEG. Nástroj je implementovaný ako single-page webová aplikácia. Užívateľ má možnosť manipulovať s grafom cez grafické užívateľské rozhranie, alebo cez textovú definíciu grafu. Je umožnené vytvárať testovacie prípady interaktívne pomocou rozhodovacej tabuľky. Nástroj podporuje automatické generovanie testovacích prípadov na základe zvoleného pokrytia. Je umožnené uloženie a spätné načítanie grafu pre editáciu. Nástroj je použiteľný na dotykových zariadeniach a na rôznych rozlíšeniach. Výstupom nástroja *ceg-editor* je report vo formáte *Markdown* v ktorom sú zhrnuté testovacie prípady, textová definícia grafu a samotný graf.

## 5.1 Rozšírenia nástroja

Možností na rozšírenie nástroja *ceg-editor* je mnoho:

- Serverová časť pre ukladanie a načítanie projektov.
- Animácie - v súčasnosti sú v nástroji všetky zmeny vykonávane okamžite, pre väčšiu užívateľskú prívetivosť by bolo vhodné tieto zmeny animovať.
- Podpora klávesových skratiek pre rýchlejšiu manipuláciu s grafom.
- Umožnenie vyjadriť grafmi časovú následnosť zahrnutím temporálnej logiky.

# Literatúra


[1] Bender RBT Inc.: *Bender RBT*. [online] http://www.benderrbt.com/

[2] MYERS, Glenford J. *The art of software testing*. 2nd ed. Hoboken, N.J.: John Wiley, c2004. ISBN 978-0471469124.

[3] MARTIN, James. *An information systems manifesto*. Englewood Cliffs, N.J.: Prentice-Hall, c1984. ISBN 978-0134647692.

[4] SMRČKA, Aleš.: *ITS - Testování na základě požadavků*. [online] https://wis.fit.vutbr.cz/FIT/st/course-files-st.php?file=%2Fcourse%2FITS-IT%2Flectures%2F8-pozadavky.pdf&cid=10020

[5] SMRČKA, Aleš.: *ITS - Testování se znalostí kódu, pokrytí logických výrazů*. [online] https://wis.fit.vutbr.cz/FIT/st/course-files-st.php?file=%2Fcourse%2FITS-IT%2Flectures%2F4-logic.pdf&cid=10020

[6] Bender RBT Inc.: *Requirements Based Testing Process Overview*. [online] http://benderrbt.com/Bender-Requirements%20Based%20Testing%20Process%20Overview.pdf

[7] The Daring Fireball Co.: *Daring Fireball: Markdown*. [online] https://daringfireball.net/projects/markdown/

[8] SVG Working Group.: *Scalable Vector Graphics (SVG) 1.1 (Second Edition)*. [online] https://www.w3.org/TR/SVG11/

[9] *JSON Schema*. [online] http://json-schema.org/

[10] MAJDA, David.: *PEG.js - Parser Generator for Javascript*. [online] https://pegjs.org/

[11] *logic-solver*.[online] https://www.npmjs.com/package/logic-solver

[12] BOSTOCK, Mike.: *D3.js - Data-Driven Documents*. [online] https://d3js.org/

[13] *Browserify*. [online] http://browserify.org/

# Príloha A

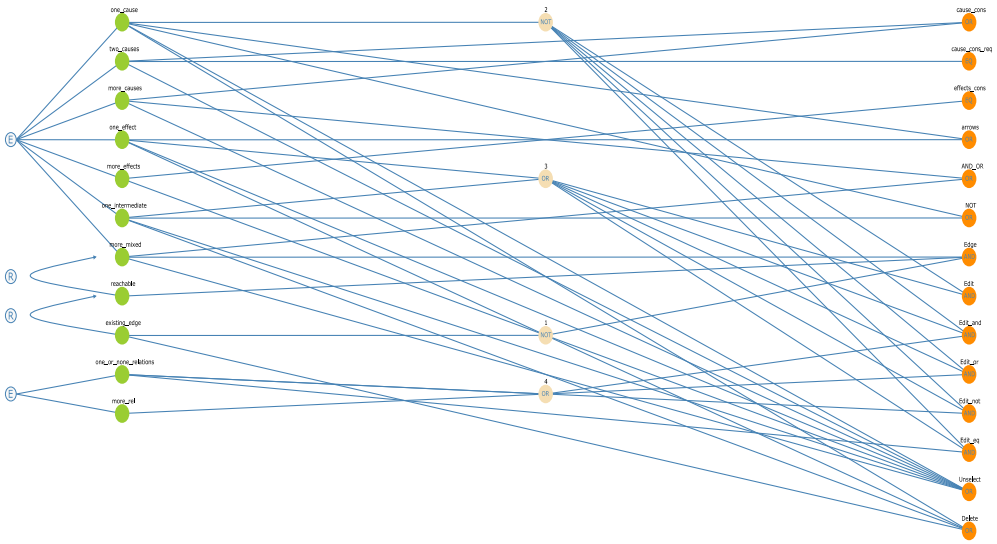
# Testovanie nástroja *ceg-editor*

Testovanie nástroja *ceg-editor* prebiehalo pomocou vytvoreného nástroja metódou testovania na základe požiadaviek. Funkcionalita testovaného nástroja bola rozdelená do čiastočne uzavretých celkov. Pre každý celok boli určené požiadavky. Následne bola každá časť transformovaná do CEG grafu. Pre menšie grafy bola vytvorená rozhodovacia tabuľka.

## Tlačidlá

V tejto časti je popísané správanie tlačidiel v hornej časti nástroja. Pre zjednodušenie, sú vynechané tlačidlá "Export", "Import", "+Causes", "+Effect", ktoré sú aktívne za každých okolnosti a tlačidlo "Undo", ktorého správanie je ťažko popísateľné CEG grafom. Požiadavky sú nasledovné:

- Ak je označená jedna príčina, alebo jeden dôsledok, tak sa vedľa uzlu zobrazia šípky, pomocou ktorý sa dajú presúvať uzly.
- Ak sú označené dve príčiny, tak je aktívne menu s obmedzeniami príčin a v ňom sú aktívne všetky položky.
- Ak je označených viac ako dve príčiny, tak je aktívne menu s obmedzeniami príčin a v ňom sú aktívne všetky položky okrem položky *requires*.
- Ak sú označené dva dôsledky, tak je aktívne menu s obmedzeniami dôsledkov a v ňom je aktívna jediná možnosť *masks*.
- Ak sú označené minimálne dva uzly a ani jeden z nich nie je dôsledok, tak sú aktívne tlačidlá "+AND" a "+OR".
- Ak je označený jeden uzol a tento uzol nie je dôsledok, tak je aktívne tlačidlo "+NOT".
- Ak sú označené dva uzly a sú dosiahnuteľné a nie je medzi nimi hrana, tak je aktívne tlačidlo "Edge".
- Ak je označený jeden uzol a tento uzol nie je príčina a má viac ako jedného predchodcu, tak je aktívne menu "Edit" a v ňom sú aktívne položky "AND", "OR".
- Ak je označený jeden uzol a tento uzol nie je príčina a má maximálne jedného predchodcu, tak sú aktívne všetky položky v menu.
- Ak je označený minimálne jeden uzol, tak je aktívne tlačidlo "Unselect".
- Ak je označený jeden uzol, tak je aktívne tlačidlo "Delete".
- Ak sú označené dva uzly a je medzi nimi hrana, tak je aktívne tlačidlo "Delete".




Obrázok A1: CEG tlačidiel

Ak je dôsledok splnený, tak je tlačidlo, ktoré reprezentuje daný dôsledok aktívne. Ak je dôsledok nesplnený, tak je tlačidlo neaktívne.

# Tlačidlá: časť 2

V tomto grafe sú popísané následky, ktoré nastanú po kliknutí na tlačidlá. Keďže sa predchádzajúci graf zaoberal okolnosťami za ktorých sú tlačidlá aktívne, tak sa v tomto grafe nespomínajú okolnosti pred stlačením. Požiadavky sú nasledovné:

- Ak je stlačená položka "Exclusive" v menu s obmedzeniami príčin, tak je pridané obmedzenie *exclusive* nad označenými uzlami.
- Ak je stlačená položka "Inclusive" v menu s obmedzeniami príčin, tak je pridané obmedzenie *inclusive* nad označenými uzlami.
- Ak je stlačená položka "One" v menu s obmedzeniami príčin, tak je pridané obmedzenie *one* nad označenými uzlami.
- Ak je stlačená položka "Requires" v menu s obmedzeniami príčin, tak je pridané obmedzenie *requires* nad označenými uzlami.
- Ak je stlačená položka "Masks" v menu s obmedzeniami dôsledkov, tak je pridané obmedzenie *masks* nad označenými uzlami.
- Ak je stlačená šípka nahor pri príčine, alebo dôsledku, tak je označený uzol posunutý nahor.
- Ak je stlačená šípka nadol pri príčine, alebo dôsledku, tak je označený uzol posunutý nadol.
- Ak je stlačené tlačidlo "AND", tak je pridaný medziuzol, ktorý je typu AND s označenými uzlami ako predchodcami.
- Ak je stlačené tlačidlo "OR", tak je pridaný medziuzol, ktorý je typu OR s označenými uzlami ako predchodcami.

- Ak je stlačené tlačidlo "NOT", tak je pridaný medziuzol, ktorý je typu NOT s označeným uzlom ako predchodcom.
- Ak je stlačené tlačidlo "+Edge", tak je pridaná hrana medzi označenými uzlami.
- Ak je stlačená položka "AND" v menu "Edit", tak je typ označeného uzla zmenený na AND.
- Ak je stlačená položka "OR" v menu "Edit", tak je typ označeného uzla zmenený na OR.
- Ak je stlačená položka "NOT" v menu "Edit", tak je typ označeného uzla zmenený na NOT.
- Ak je stlačená položka "EQ" v menu "Edit", tak je typ označeného uzla zmenený na EQ.
- Ak je stlačené tlačidlo "Unselect", tak sa zruší označenie všetkých uzlov.
- Ak je stlačené tlačidlo "Delete", tak je zmazaný uzol, alebo hrana.

Obrázok A2: CEG akcí tlačidiel

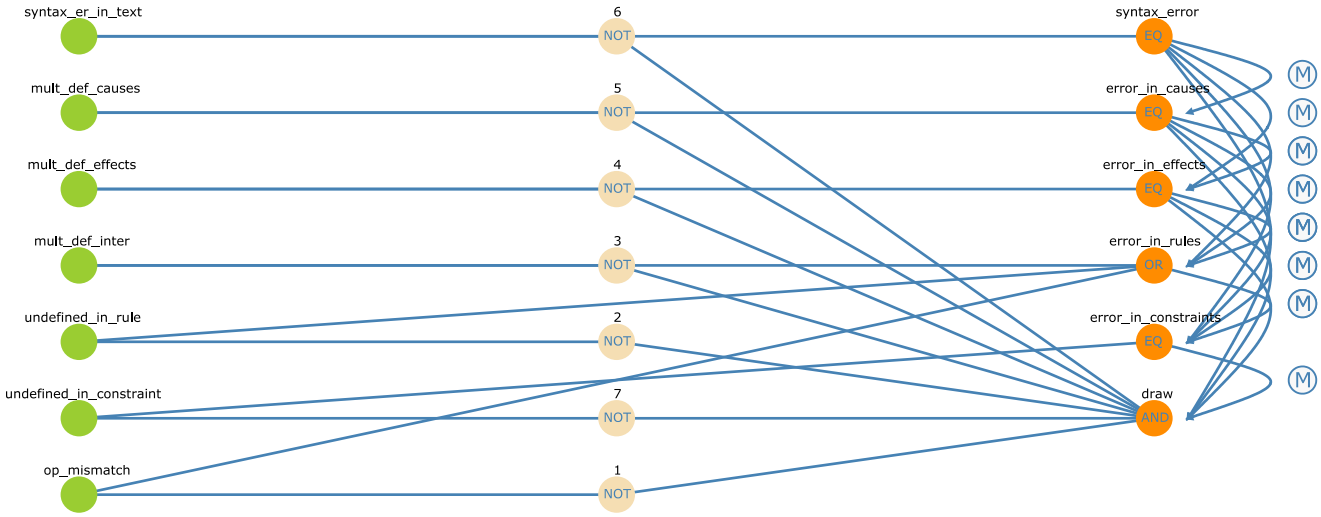
Kliknutie na tlačidlá je reprezentované príčinami, odpovedajúca akcia dôsledkami.

# Textová definícia

V tejto časti sú popísané situácie, ktoré môžu nastať pri manipulácií s textovou definíciou. Z veľkej časti ide o vysporiadanie sa so syntaktickými a sémantickými chybami, ktoré môžu pri zadávaní textovej definície nastať. Požiadavky sú nasledovné:

- Ak nastane počas spracovania syntaktická chyba, tak je vypísaná syntaktická chybová správa.
- Ak dôjde ku viacnásobnej definícií pri príčinách, tak je vypísaná hláška o chybe v príčinách.

- Ak dôjde ku viacnásobnej definícií pri dôsledkoch, tak je vypísaná hláška o chybe v dôsledkoch.
- Ak dôjde ku viacnásobnej definícií medziuzlu, tak je vypísaná hláška o viacnásobnej definícií.
- Ak sa na pravej strane pravidla vyskytne nedefinovaný identifikátor, tak je vypísaná hláška o tomto identifikátore.
- Ak sa v definícií obmedzenia vyskytne nedefinovaný identifikátor, tak je vypísaná chybová hláška o chybe v definícií obmedzení.
- Ak sa v pravidle vyskytne viac operátorov, tak je vypísaná chybová hláška o výskyte viacerých operátorov.
- Ak je textová definícia spracovaná bez akejkoľvek chyby, tak je graf vykreslený.
- Ak sa vyskytne v definícií chyba, tak ďalšie vyskytujúce sa chyby sú neodhalené.




Obrázok A3: CEG manipulácie s textovou definíciou

# Import/export grafu

Časť zaoberajúca sa načítaním grafu zo súboru a uložením grafu do súboru. Požiadavky sú nasledovné:

- Ak si užívateľ zvolí exportovať celý graf s textovou definíciou, testovacími prípadmi a grafom, tak by sa malo exportovanie podariť.
- Ak si užívateľ zvolí exportovať len testovacie prípady, tak by sa mali exportovať úspešne.
- Ak si užívateľ zvolí exportovať samotný graf, tak by sa mal úspešne exportovať.
- Ak si užívateľ zvolí exportovať report v Markdowne, tak by sa export mal úspešne podariť.
- Ak je importovaný validný súbor s grafom, tak by sa mal načítať a nový graf vykresliť.
- Ak je importovaný nevalidný súbor, tak by malo byť užívateľovi ohlásené, že súbor bol nevalidný.


Obrázok A4: CEG načítavania a ukladania grafu

| Name | Description | [1] | [2] | [3] | [4] | [5] | [6] | [7] |
|---|---|---|---|---|---|---|---|---|
| export_whole | export whole project | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| export_tc | export test cases only | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| export_svg | export svg only | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| export_report | export report only | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| import_valid | import valid file | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| import_invalid | import invalid file | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| exported_whole | whole graph was exported | true | false | false | false | false | false | false |
| exported_tc | test cases were exported | false | false | true | false | false | false | false |
| exported_svg | svg was exported | false | false | false | true | false | false | false |
| exported_report | report was exported | false | false | false | false | false | true | false |
| imported_valid | file was imported | false | false | false | false | true | false | false |
| error | error caused by invalid file | false | false | false | false | false | false | true |

Tabuľka A1: Rozhodovacia tabuľka CEG export/import

# Rozhodovacia tabuľka: časť 1

Nasledujúce tri časti sa zaoberajú rozhodovacou tabuľkou a manipuláciou s ňou. Požiadavky sú nasledovné:

- Ak užívateľ klikne na tlačidlo "+", tak je do tabuľky pridaný stĺpec s voľnými bunkami
- Ak po stlačení tlačidla "+" je kliknuté na bunku v tabuľke, tak je označený uzol, prislúchajúci bunke.
- Ak užívateľ zmení popis príčiny, alebo dôsledku v tabuľke, tak je táto zmena odrazená v textovej definícií.
- Ak je zmenený názov testovacieho prípadu, tak bude tento názov aktualizovaný.
- Po pridaní príčiny je pridaný riadok s novou príčinou do tabuľky.
- Po pridaní dôsledku je pridaný riadok s novým dôsledkom do tabuľky.
- Po zmazaní príčiny je zmazaný aj riadok, ktorý príčine prislúcha
- Po zmazaní dôsledku je zmazaný aj riadok, ktorý dôsledku prislúcha.


Obrázok A5: CEG rozhodovacej tabuľky č.1

V tomto grafe je znázornená následnosť udalostí. Na to aby mohol užívateľ kliknúť na bunku v tabuľke musí najprv pridať stĺpec tabuľky. Teda na to aby bola splnená príčina *a2_click_on_cell* musí byť najprv splnená príčina *a1_plus_button*. Preto je medzi týmito príčinami relácia *requires*. Týmto spôsobom je napodobnená časová následnosť.

# Rozhodovacia tabuľka: časť 2

Požiadavky sú nasledujúce:

- Ak užívateľ klikne na tlačidlo "+", tak je pridaný stĺpec v tabuľke.

- Ak užívateľ klikne na editovateľnú bunku, tak je v hlavičke stĺpca vytvorené tlačidlo "-"
- Ak užívateľ klikne na tlačidlo "-", tak je daný stĺpec zmazaný.
- Ak je stĺpec vyplnený hodnotami, tak sú vyhodnotené dôsledky a výsledky vypísané v tabuľke.
- Ak vyplnené hodnoty stĺpca nevyhovujú nejakému obmedzeniu, tak je v hlavičke pridané tlačidlo "Show".
- Ak je kliknuté na tlačidlo "Show", tak sú označené obmedzenia, ktorým hodnoty príčin nevyhovujú.
- Ak užívateľ stlačí klávesu TAB keď sa nachádza v editovateľnej bunke, tak je kurzor posunutý na ďalšiu bunku v stĺpci.


Obrázok A6: CEG rozhodovacej tabuľky č.2

| **Name** | **Description** | **[1]** | **[2]** | **[3]** |
|---|---|---|---|---|
| plus_button | plus button is pressed | 0 | 1 | 1 |
| click_cell | user clicks on editable cell | 0 | 1 | 1 |
| minus_button | minus button is pressed | 0 | 1 | 0 |
| fill_column | user fills column | 0 | 0 | 1 |
| constraint_violation | filled values violates constraint | 0 | 0 | 1 |
| show_button_clicked | show button clicked | 0 | 0 | 1 |
| tab_pressed | tabulator pressed | 0 | 1 | 0 |
| case_added | test case (column) added | false | true | true |
| button_shown | minus button shown | false | true | true |
| case_deleted | test case deleted | false | true | false |
| evaluate_effects | effects are evaluated | false | false | true |
| show_button_present | 'show' button present | false | false | true |

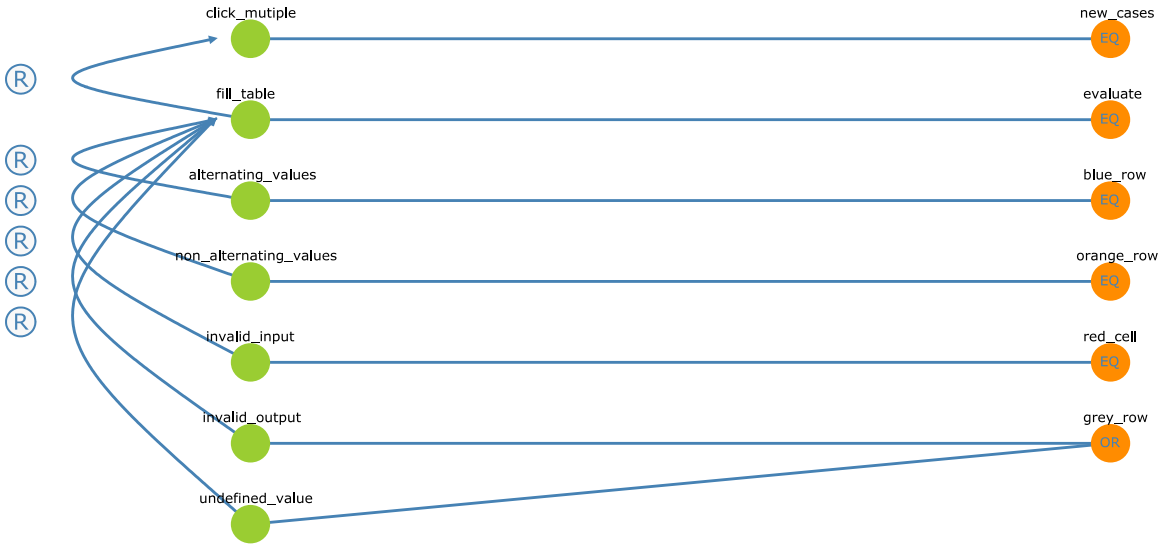
| | | | | |
|---|---|---|---|---|
| constraints_marked | constraints that are violated are marked | false | false | true |
| shift_to_next_cell | focus is shifted on the next cell | false | true | false |

Tabuľka A2: Rozhodovacia tabuľka CEG "Rozhodovacia tabuľka: časť 2"

# Rozhodovacia tabuľka: časť 3

Požiadavky sú nasledovné:

- Pokiaľ je na tlačidlo "+" kliknuté viackrát, tak sú vytvorené viaceré testovacie prípady.
- Keď sú vyplnené viaceré stĺpce, tak vyhodnotené dôsledky sa zapíšu do príslušných stĺpcov.
- Ak alternujú hodnoty v riadku, tak je daný riadok podfarbený na modro.
- Ak nealternujú hodnoty v riadku, tak je daný riadok podfarbený na oranžovo.
- Ak je v bunke prislúchajúcej príčine zadaná chybná hodnota, tak je bunka podfarbená na červeno.
- Ak je v bunke prislúchajúcej dôsledku nedefinovaná, alebo neznáma hodnota, tak je riadok podfarbený na šedo.




Obrázok A7: CEG rozhodovacej tabuľky č.3

| Name | Description | [1] | [2] | [3] | [4] |
|---|---|---|---|---|---|
| click_mutiple | user clicks on 'plus' button multiple times | 0 | 1 | 1 | 1 |
| fill_table | user fills table with values | 0 | 1 | 1 | 1 |

| | | | | | |
|---|---|---|---|---|---|
| alternating_values | values in row are alternating | 0 | 1 | 1 | 1 |
| non_alternating_values | values in the row are same | 0 | 0 | 0 | 1 |
| invalid_input | user enters invalid input in the cell | 0 | 1 | 0 | 0 |
| invalid_output | effect is evaluated with wrong value | 0 | 1 | 0 | 0 |
| undefined_value | effect with undefined value | 0 | 0 | 1 | 0 |
| new_cases | new cases are produced | false | true | true | true |
| evaluate | effects are evaluated | false | true | true | true |
| blue_row | row with alternating values is blue | false | true | true | true |
| orange_row | row with non alternating values is orange | false | false | false | true |
| red_cell | cell with error is red and row with error is white | false | true | false | false |
| grey_row | row with undefined value or with wrong output is grey | false | true | true | false |

Tabuľka A3: Rozhodovacia tabuľka grafu "Rozhodovacia tabuľka časť: 3"

# Príloha B

# Príklad vygenerovaného reportu vo formáte Markdown

````
# Text Definition

## Cause-Effect Graph

![text](Project_name.svg)

### Causes:

```
1: The first character is 'A'
2: The first character is 'B'
3: The second character is a digit
```

### Effects:

```
71: Exception X12
70: Update of a file
72: Exception X13
```

### Rules:

```
11 = 1 || 2
71 = !11
70 = 11 && 3
72 = !3
```

### Constraints:
````

```
E: 1, 2
71 masks 72
```

# Test Cases

### [1]

```
1: 0
2: 0
3: 1
71: true
70: false
72: false
```

### [2]

```
1: 0
2: 1
3: 1
71: false
70: true
72: false
```

### [3]

```
1: 1
2: 0
3: 1
71: false
70: true
72: false
```

### [4]

```
1: 1
2: 0
3: 0
71: false
70: false
72: true
```
### [5]

```
1: 0
2: 0
3: 0
71: true
70: false
72: false
```